CellPress

Trends in Cognitive Sciences

Review

Memory as a Computational Resource

Ishita Dasgupta[1,4,*] and Samuel J. Gershman[2,3]

Computer scientists have long recognized that naive implementations of algorithms often result in a paralyzing degree of redundant computation. More sophisticated implementations harness the power of memory by storing computational results and reusing them later. We review the application of these ideas to cognitive science, in four case studies (mental arithmetic, mental imagery, planning, and probabilistic inference). Despite their superficial differences, these cognitive processes share a common reliance on memory that enables efficient computation.

Highlights

Most computations that people do in everyday life are very expensive. Recent research highlights that humans make efficient use of their limited computational resources to tackle these problems.

Memory is a crucial aspect of algorithmic efficiency and permits the reuse of past computation through memoization.

We review neural and behavioral evidence of humans reusing past computations across several domains, including mental imagery, arithmetic, planning, and probabilistic inference.

Recent developments in neural networks expand the scope of computational reuse with a distributed form of memoization called amortization. This opens many new avenues of research.

The Efficiency of Computational Reuse

Many computational problems would be impossible to solve without the use of memory. Consider, for example, the problem of finding the shortest path between two locations. Naively, you might try solving this problem by brute force, enumerating all possible n-step paths. If at each step you can move in one of k directions, then you would have to enumerate k^n paths, an astronomical number for even small n (for just ten steps and four directions, there are over one million possible paths). Fortunately, the problem has internal structure that can be exploited: some paths are subcomponents of longer paths, so you can reuse computation of the subpaths when computing the longer path. If you already know the shortest paths from A to B and B to C, you can combine them to find the shortest path from A to C going through B; all other paths touching these three locations can be ignored. To apply this strategy, it is necessary to store information about subpaths so that they can be reused later. Path planning is an example of how the recursive structure of a problem can be used to implement a **space–time trade-off** (see Glossary): redundant computation (which costs time) can be avoided by storing partial solutions in memory (which costs space) [1].

The idea that responses can be stored in memory and reused to answer future queries has a long history in cognitive science. Caching is implicit in the work of De Groot [2] and Newell *et al*. [3] and has been advocated more explicitly by Logan and his colleagues [4]. A rather different (but equally ubiquitous) role for memory has been developed in production system architectures like ACT-R [5] and SOAR [6], where memory serves to strengthen and link together elementary computations (productions) into more complex cognitive skills. We discuss these cognitive models of computational reuse in Box 1. Modern computer science approaches to computational reuse go beyond these models to provide practical solutions to the issues of generalization, persistence, and decomposition, as we discuss in the next section.

We begin this review with a general overview of computational reuse strategies. We then examine four domains of cognitive science in which these strategies have been studied: mental arithmetic, mental imagery, probabilistic inference, and planning.

Approaches to Computational Reuse

In an influential paper [7], the computer scientist Donald Michie proposed **memoization** as a technique to reduce redundant computation. Function calls are intercepted by a 'memoizer' that inspects a cache (the memo-table) of past function calls and their outputs. If a function has previously been called with the same inputs, then the previously computed result is reused. If

[1]Department of Computer Science, Princeton University, Princeton, NJ, USA
[2]Department of Psychology, Center for Brain Science, Harvard University, Cambridge, MA, USA
[3]Center for Brains, Minds, and Machines, MIT, Cambridge, MA, USA
[4]Current address: DeepMind, New York City, NY, USA.

*Correspondence: idg@google.com (I. Dasgupta).

Box 1. Comparing Cognitive Models of Reuse

Logan's instance theory of automaticity [4] exemplifies the top-down approach to computational reuse: function outputs (problem solutions) are stored with their inputs (stimuli) and retrieved whenever a new input matches the stored input. This form of reuse is top-down in the sense that the output of the most complex function call is stored and reused; all intermediate computations are forgotten. This has the advantage, shared with classical memoization techniques, that it is simple to implement and can be applied very generally, but also shares the disadvantage of failing to exploit the internal structure of complex computations. For example, many probabilistic inference algorithms exploit conditional independence (e.g., the Markov property; Box 2) to break complex problems down into simpler subproblems. Stored solutions to these subproblems can then be used to assemble solutions to the complex problems.

Production system architectures such as ACT-R [89,90] and SOAR [6] exemplify the bottom-up approach to computational reuse: intermediate computations (productions) can be strengthened, making them more likely to be activated in the future. They can also be compiled to form more complex computations (chunks) that are then subject to strengthening. This approach has the advantage that it can flexibly reuse intermediate computations when this is advantageous. In practice, this form of reuse can be tricky to implement effectively. Storing intermediate computations means that there is a very large database, placing onerous demands on storage and retrieval [91].

Another issue for reuse in production systems concerns the normative assumptions underlying strengthening procedures. In the ACT-R architecture [89], strengthening is based on the marginal probability of evoking a production or chunk. This implicitly assumes that the posterior over production/chunk sequences is factorized, similar to mean-field approximations in variational inference [92]. This assumption is computationally convenient because it avoids the problem of computing a posterior over the combinatorial space of sequences. However, the cost is that such an approximation can be highly inaccurate.

the target function itself has not been memoized, any subfunctions that have been memoized can be intercepted as the target function is executed.

We can use the classic example of the Fibonacci series to illustrate these ideas. The Fibonacci series is defined recursively as the sum of the previous two function values:

$$f(n) = f(n-1) + f(n-2), \quad [1]$$

with initial conditions $f(0) = 0$ and $f(1) = 1$. Implemented naively, you get a computation graph shown in Figure 1A, where the same function is called multiple times with the same input. Memoization works by wrapping $f(n)$ in a memoizer, which consults a memo-table each time the function is called to determine whether it has been called before, in which case the cached output can be reused (Figure 1B).

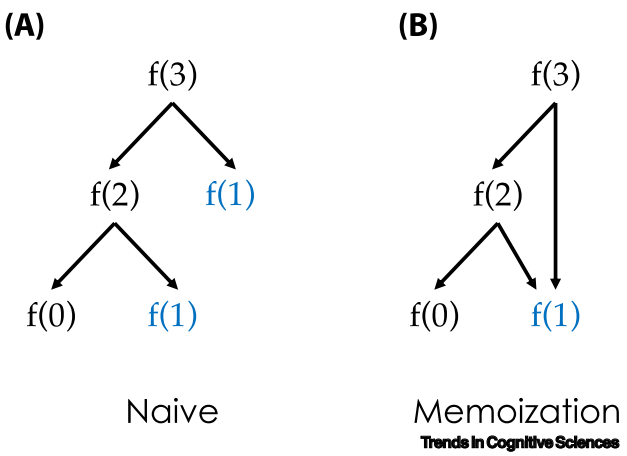

Figure 1. Computation Graphs for Calculating the Fibonacci Series. (A) Naive implementation, without computational reuse. Redundant function calls are highlighted in blue. (B) Memoization, in which function outputs are added to a cache (the memo-table) for reuse. The top-down flow of computation mirrors the naive implementation, except that function calls are intercepted by the memo-table.

Glossary

Amortized inference: memoizing the process of inference, by learning a mapping from data to the parameters of the posterior distribution.
Bellman equation: a recursive formulation of the value function for an MDP.
Hidden Markov model: a generative sequence model consisting of two components: a transition distribution that samples the next hidden state based on the current one, and an observation distribution that samples an observation conditional on the hidden state.
Markov decision process (MDP): a mathematical framework to model sequential decision making. It consists of a set of states, a set of actions, the probability that action a in state s leads to state s', and the reward received when taking action a in state s. Rewards and transitions are conditionally independent of history given the current state and action.
Memoization: storing the results of expensive function calls and returning the cached result when the same inputs occur again, in order to speed up algorithms or programs.
Space–time trade-off: where an algorithm trades increased time usage with decreased space usage and vice versa. This is sometimes also called a time–memory trade-off.
Sufficient statistic: a summary of a data set that completely determines the parameters governing the data distribution.
Temporal difference learning: approximates value iteration when the Bellman equation cannot be computed (the expectation over states is unavailable or intractable) by sampling from the environment and incrementally updating value estimates based on each sample.
Value iteration: solves the Bellman equation for an MDP by updating value estimates to locally satisfy the Bellman equation until convergence.

Practical memoization schemes must address a number of issues. One is generalization. Michie recognized that a memo-table might be too rigid for some problems, particularly when the input space is vast and the probability of a repeated function call is low. He proposed to solve this problem by interpolating between cached outputs. For example, $f(n)$ is monotonically increasing in n, so $f(n)$ must lie between $f(n-1)$ and $f(n+1)$. In fact, Johannes Kepler observed that the ratio between consecutive Fibonacci numbers converges to a constant (the golden ratio), which implies that for large n the output is the geometric mean of the outputs for the neighboring inputs:

$$f(n) \approx \sqrt{f(n+1)f(n-1)}. \quad [2]$$

Thus, Fibonacci numbers can always be approximated by geometrically interpolating neighboring numbers, if these are in the memo-table. When the queries take the form of natural numbers, similarity is clearly defined, making interpolation easy. Gauging query similarity might not be as obvious for other domains.

A second issue concerns persistence. Because there are inevitably space constraints, memoization must be selective about what it stores and when it discards information [8,9]. At one extreme (minimal space cost), the memo-table is erased after each function call. At the other extreme (maximal space cost), the memo-table persists indefinitely across function calls. Many of the psychological phenomena we consider later require persistence beyond a single function call, but it is an open question how long these completed computations persist in memory and in what form.

A third issue concerns decomposition strategy. The Fibonacci example above illustrates a 'top-down' strategy in which subfunctions are cached 'lazily' whenever they are called; in this way, more complex functions are invoked before simpler ones. An alternative 'bottom-up' decomposition strategy (sometimes referred to as tabulation [10]) is to call the simpler functions first and then build up to the complex functions.

Four Case Studies

In the following subsections, we discuss computational reuse as a cognitive mechanism in four domains. Despite their many differences, these domains have in common the property that they benefit from algorithmic efficiency via problem decomposition and partial reuse. Empirical findings from these domains show that training strengthens memory for computational results. We first consider the domains of mental arithmetic and imagery, where existing cognitive models have invoked caching and reuse. We then discuss more recent work applying these ideas to planning and probabilistic inference. Machine learning has long studied these problems to find computationally feasible solutions that invoke sophisticated forms of reuse. We review these approaches and how they might inform our understanding of cognitive mechanisms.

Arithmetic

When children learn arithmetic, they appear to rely on counting routines. For example, when given single digit addition problems (e.g., 4 + 5), they initialize a counter at the larger addend (5) and then increment it by 1 at a roughly constant rate until the number of increments equals the smaller addend (4). The strongest evidence for this 'min-count' model comes from chronometric data showing that response times increase linearly with the minimum addend [11]. By the age of 10 years, most children appear to rely primarily on a different strategy, retrieving addition facts directly from memory, as evidenced by a dramatic flattening of the response time function [12–14]. A similar shift is observed in adults trained to perform alphabet arithmetic (e.g., $A + 2 = C$). Due to the unfamiliarity of these problems, adults are effectively placed in a situation similar to the child first learning arithmetic and transition with experience from counting to memory retrieval (Figure 2A) [15,16].

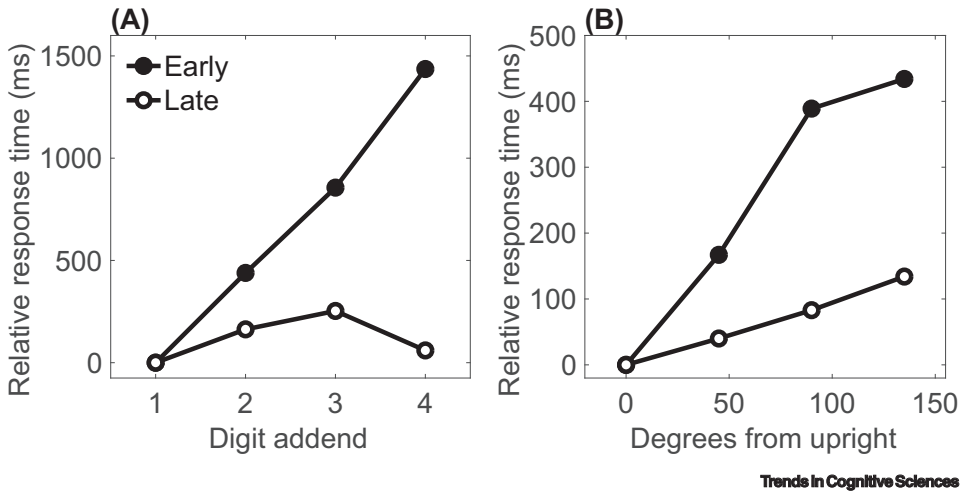

Figure 2. Response Time Functions for Alphabet Addition (A) and Mental Rotation (B). To highlight the change in slope as a function of practice (early versus late), the functions have been plotted relative to the response time for the minimum x-axis value. Data in (A) replotted from [15]; data in (B) replotted from [17].

This shift from algorithm to memory was elevated to a general theory of cognitive skill acquisition by Gordon Logan in his 'instance theory of automaticity' [4] (see Box 1 for comparison with other models). The key idea is that each time an algorithm is called, its output is stored in memory and can be potentially reused later. Logan formalized the competition between algorithm and memory as a race, whereby behavioral outputs were based on the first process to complete. With practice, more memory traces are available and hence it becomes more likely that one of them will win the race. The instance theory of automaticity is a theory of memoized cognition: it posits that reuse operates 'top-down' by storing the outputs of completed computations in a memo-table (long-term declarative memory).

There is also some evidence for 'bottom-up' strategies in mental arithmetic. For example, when faced with multidigit arithmetic problems (e.g., 31 + 25), older children often adopt a decomposition strategy in which the addends are split into tens and units (e.g., 31 is split into 30 and 1), so that the multidigit problem can be solved by simple addition (31 + 25 = 30 + 20 + 1 + 5) [18–20]. This decomposition is not easily addressed by existing top-down race models (e.g., Logan's model) that store the final response for future reuse without any decomposition into intermediate computations.

Imagery

The study of mental imagery parallels the study of mental arithmetic reviewed in the previous section. In a classic experiment, Shepard and Metzler [21] asked subjects to judge whether two line drawings depicted the same 3D object. On trials in which the drawings depicted rotated versions of the same object, response times increased linearly with the angular difference, consistent with the hypothesis that subjects were mentally rotating the object much as they would a physical object. These findings are analogous to the response time functions for mental arithmetic, suggesting an underlying iterative accumulation algorithm (counting or rotating). Imagery also increasingly relies on memory over the course of training [17,22]. Tarr and Pinker [17] found that response time functions for mental rotation flatten considerably after extensive practice (Figure 2B). This again fits with a top-down instance-based memoization theory of the form proposed by Logan [4].

Also similar to arithmetic, effective memoization in this domain requires us to go beyond the reuse of individual instances. For example, considerable evidence from studies of motion perception suggest that we represent multipart objects using a hierarchy of relative coordinate frames [23–25], including the dependency structure of scenes and objects at each level. This permits a form of partial reuse and bottom-up memoization: the relative poses of each part can be computed and cached for reuse in a whole-object transformation [26]. This suggests that the unit of computational reuse is not whole instances; humans learn representations that permit more effective partial reuse and generalization.

Probabilistic Inference

Perception is frequently faced with ambiguity: the size, shape, brightness, and color of objects are not uniquely resolved by their sensory traces. Higher-level cognitive processes like language understanding and social inference are similarly hampered by ambiguity. Probability theory offers a self-consistent framework for reasoning under ambiguity and has served as a useful normative benchmark for understanding how the brain deals with ambiguity [27–29].

Consider an agent who has observed data d (e.g., a doctor is told that a patient has a cough). There are multiple latent variables (denoted by z) that could potentially have produced d. For example, the cough could have been generated by the common cold, or by lung cancer. Some values of z are more likely to have produced d than others (in this example, lung cancer produces coughing more reliably than the common cold). In addition, some values of z are more likely to occur *a priori* than others (the common cold is more likely than lung cancer). Probability theory tells us that the optimal inference takes the form of a posterior probability distribution, as prescribed by Bayes' rule:

$$P(z \mid d) = \frac{P(d \mid z)P(z)}{\sum_{z'} P(d \mid z')P(z')}, \quad [3]$$

where $P(d \mid z)$ is the likelihood of the data under hypothesis z and $P(z)$ is the prior. Intuitively, the likelihood captures the fit between a hypothesis and data, and the prior captures knowledge about the frequency of occurrence for a hypothesis (although probability theory can also accommodate hypotheses that only exist in our minds rather than in the external world).

The fundamental problem for any resource-limited agent is that Bayes' rule is intractable in the general case. This problem arises ubiquitously, for example, any time there is a combinatorial hypothesis space. In the clinical diagnosis example, there may be multiple nonexclusive conditions (e.g., a person may have the common cold and lung cancer at the same time). Thus, combinatorial structure runs into the same kind of exponential explosion in time complexity found in planning problems. Fortunately, just as agents can harness the Markov property in the service of planning (see next section), they can also harness the Markov property in the service of probabilistic inference, breaking complex inference problems into simpler ones (Box 2). We can recognize this strategy as a form of bottom-up computational reuse of solutions to subproblems in order to avoid redundant computation.

A classic example of cached inference is the Kalman filter, which implements exact inference in a linear-Gaussian hidden Markov model (a special case of the model described in Box 2). In part

Box 2. Breaking Down Inference Problems Using the Markov Property

For illustration, we analyze a simple model in which Bayes' rule is actually tractable, despite the combinatorial structure. Suppose an agent observes a data sequence $d_{1:n} = [d_1, \ldots, D_n]$ and the agent assumes that this sequence was generated by a corresponding sequence of latent states $z_{1:n} = [z_1, \ldots, z_n]$. If the agent wants to compute the posterior over the state at time n, then probability theory dictates that it has to marginalize over all possible state histories (i.e., all the different ways it could have arrived at z_n):

$$P(z_n \mid d_{1:n}) = \sum_{z_{1:n-1}} P(z_n, z_{1:n-1} | d_{1:n}) \quad [\mathrm{I}]$$

In other words, the agent would have to do inference over the combinatorial hypothesis space of n-step state sequences. If the latent variables are continuous, the summation is replaced with integration, which makes the problem potentially even more challenging.

Critically, if the agent also assumes that the generative process is a **hidden Markov model**, in which observations and transitions between states depend only on the current state, then it can break the problem down as follows:

$$P(z_n | d_{1:n}) = \sum_{z_{n-1}} P(z_n | d_n, z_{n-1}) P(z_{n-1} | d_{1:n-1}). \quad [\mathrm{II}]$$

Notice that now the agent only needs to marginalize over the last state, not the whole sequence. Furthermore, $P(z_{n-1} \mid d_{1:n-1})$ is simply the posterior from the last time step. This means that the equation can be applied iteratively at each time step to maintain the exact posterior distribution. The basic idea of using Markov assumptions to break a complex inference problem down into simpler problems appears in many different contexts, including more challenging inference problems [92].

The implication for memory is that only the sufficient statistics of the posterior need to be stored in order to implement probabilistic inference and these statistics can be updated based on the currently available data (d_n) without retrieving past data. Thus, inference algorithms can reduce their computational footprint by exploiting the structure of the probabilistic model and storing the appropriate intermediate computations.

because of its resemblance to nonprobabilistic error-driven learning rules, the Kalman filter has been proposed as a psychologically plausible model of learning [30–32]. Moreover, it can be implemented in biologically plausible neural circuits [33–35]. What makes the Kalman filter efficient is that it stores intermediate results (the posterior **sufficient statistics**) in a cache and updates them in closed form at the next time step, thereby bootstrapping intermediate computations. This keeps the computational cost of inference bounded even when dealing with an infinite stream of data.

While the kind of caching strategy described earlier is a powerful tool for exact inference, there are many probabilistic models in which it is either still too expensive (e.g., if the sufficient statistics are not low-dimensional) or cannot be applied at all (e.g., if the marginalization lacks a closed form solution). One approach to this problem is to follow in the spirit of the Kalman filter, but instead approximate these computations. This can be achieved using Monte Carlo sampling [36], variational approximation [37], as well as combinations of the two [38,39] that utilize their complementary benefits. These generalizations of the Kalman filter have been used to model human probabilistic inference in sequential tasks, like multiple object-tracking [40] and sentence comprehension [41], that do not have simple closed form solutions. However, these approximate computations can also be very expensive: Monte Carlo sampling requires many samples to give good estimates and variational inference requires a sometimes expensive optimization in order to get good approximations. An alternative is to instead use a top-down approach that directly caches inferences from previous experience.

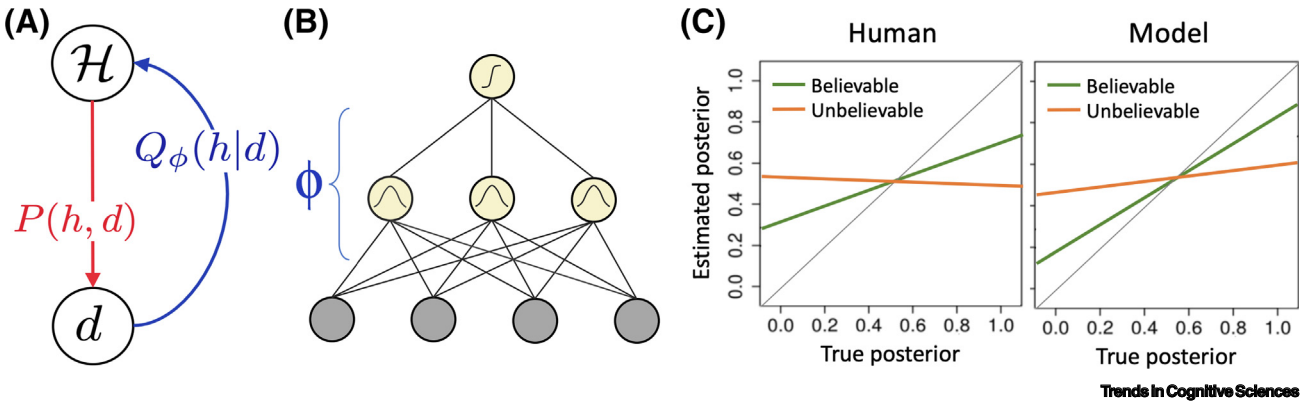

Figure 3. Amortizing Posterior Inference. (A) $P(h, d)$ is a generative model that produces latent variables h and data d given h, $Q_\phi(h \mid d)$ is a recognition model, parametrized by ϕ, that maps observable data to the posterior distribution over underlying hypotheses. (B) The recognition network used in [51], its capacity depends on the number of nodes in the middle layer. (C) Humans are better at Bayesian inference when the provided probabilities are believable (replotted from [52]). The same pattern arises in a recognition network with limited capacity (replotted from [51]).

A resurgence of interest in this kind of top-down memoization for probabilistic inference was recently kindled by the development of the variational auto-encoder, a neural network that maps data inputs to an approximate posterior [42]. In essence, the neural network functions as a distributed memo-table, obviating the need for bottom-up iterative computation. The variational auto-encoder is an example of a more general family of algorithms, known as **amortized inference**, which replace iterative computation with a fast parametrized 'recognition model' [43–46]. The parameters of this recognition model can be learned concurrently with the parameters of the generative model. When the recognition model takes the form of a neural network parametrized by synaptic weights, we refer to it as a 'recognition network'. Such a network is a sophisticated version of Michie's interpolation approach to memoization, able to operate on inputs it has never seen before by generalizing from past inputs. The strengths and shortcomings of top-down memoization with recognition networks can be traced back to the strengths and shortcomings of neural networks in general. They are very expressive and can interpolate well even in complex and high dimensional inference problems like visual recognition and natural language processing. However, as they currently stand, they require a lot of data and struggle to generalize compositionally [47,48].

Recent work in psychology and neuroscience has explored whether the brain may be using amortized inference. Yildirim and colleagues [49] have proposed that the ventral visual stream linking primary visual cortex to the medial temporal lobe implements a fast recognition network for object and scene perception. Applying this proposal to face perception, they showed how such a model could efficiently compute inferences about 3D form and face identity, matching human robustness to variation in lighting and pose [50]. They also showed that the internal representations encoded by the recognition network matched population activity of neurons recorded in macaque ventral visual stream during face processing.

In another line of work [52,53], we have tested a key implication of amortized inference: past inferences should exert an influence on future inferences, because they imprint themselves on the parameters of the recognition model. We demonstrated experimentally that inferences are biased by recent inferences computed for other queries and that this bias depends on

the overlap between the queries. To explain such biases, we showed how amortization using a neural network leads to sharing of structure between different posteriors, provided the network is capacity-limited (e.g., with a bottleneck in the mapping from input to output) [51] (Figure 3). Such sharing can also explain a number of biases reported in the psychology literature. For example, people are much more accurate at Bayesian inference when the probabilities are realistic [54,55]. This finding is at odds with the view that the brain relies solely on a general-purpose inference engine that operates on arbitrary probabilities. By contrast, our theory accommodates this finding by assuming that the recognition network preferentially allocates its limited capacity to answering high probability queries. This preferential allocation is an automatic consequence of training the recognition network on queries drawn from the environment. We further showed that experimentally manipulating the historical query distribution can shift the pattern of bias, consistent with the preferential allocation hypothesis.

Planning

Planning a sequence of actions to maximize cumulative reward is challenging because the number of possible sequences explodes combinatorially with the planning horizon. With only two actions, an agent would have to compare over 1000 different action sequences just to plan over a ten-step horizon. An archetypal example of such planning is in games like chess and Go. Understanding this kind of planning, both how humans do it as well as how to artificially replicate it, has historically played a central role in both the study of artificial intelligence as well as cognitive science [2,3,56,57].

Exact planning requires that we evaluate every possible game tree and then choose the best one. Several empirical studies have found that humans do not perform this kind of exact planning and often evaluate game states based on memory of familiar board positions from previous games [2,58–60]. These studies provide evidence that humans rely on memoization and partial reuse for efficient planning, but still leave many questions unanswered about the precise mechanism of reuse. In the following section we consider various computational implementations of memoization for planning, their use in modern machine learning, as well as their cognitive and neural signatures.

To simplify the problem, we first outline a standard set of assumptions made about the structure of the environment, known as a **Markov decision process (MDP)**. An agent can move between states by selecting actions and collecting rewards; critically, the rewards and transitions depend only on the current state and action. This is known as the Markov property and allows us to write the expected cumulative reward (value) in a recursive form known as the **Bellman equation** [61]:

$$Q(s,a) = R(s,a) + \mathbb{E}[Q(s',a')], \qquad [4]$$

where s denotes the current state, a denotes the action, $R(s, a)$ is the expected reward, and the expectation $\mathbb{E}[Q(s',a')]$ denotes an average of the value function over the next step state (s') and action (a'). In some cases a discount factor multiplies the second term, to capture down-weighting of distal rewards. The Bellman equation underwrites most efficient planning algorithms, because it lets the planner decompose a complex problem into a series of simpler ones. Specifically, an agent can iterate over states and actions, recomputing the action that maximizes the right-hand side, and this procedure (known as **value iteration**) is guaranteed to converge to the optimal policy in a finite number of iterations.

To compute the expectation in the Bellman equation, the agent needs access to a model of the environment, in particular the transition function that specifies the probabilities of each possible next state given the current state and action. In some domains, this function is not available,

but the agent can still find the optimal policy by interacting with the environment and using samples from these interactions to approximate the expectation (Equation 4). This leads to a family of 'model-free' reinforcement learning algorithms, the most prominent of which is known as **temporal difference learning** [62]. This can still be challenging in domains that contain many possible states, resulting in a large and unwieldy memo-table for the Q-value of every state-action pair. The key innovation of deep reinforcement learning [63,64] is to instead use a neural network to learn this mapping. Similar to the amortized recognition network from the previous section, a Q-network operates as a distributed memo-table, permitting partial reuse across similar inputs and generalization to new inputs.

When we have access to an internal model but calculating the expectation is too expensive, we can use the internal model as a simulator, generating fictive experience 'offline' that can then be cached into a temporal difference learning algorithm [62]. These simulations themselves are also expensive and can be intelligently allocated by prioritizing the exploration of states and actions with high cached values [65]. This strategic use of a model in conjunction with temporal difference learning was a crucial part of the recent successes in superseding human performance in Go [66].

The model-based and model-free algorithms described earlier have in common their reliance on a bottom-up decomposition approach: subproblems are solved and then cached for reuse in solving larger problems. These algorithms have been highly successful as theories of how the brain solves sequential decision-making problems [67]. Caching can explain why humans and other animals sometimes make decisions based on 'stale' information, choosing actions that bring them to previously high-value states that have been devalued [68–71], for example, when rodents press a lever to obtain food on which they have been satiated and hence do not want to consume. This pattern of behavior arises if some parts of the cache have not yet been updated after other parts have been updated (inducing inconsistencies between the cached values). Partial cache updating is an inevitable consequence of a resource-constrained planning system that must prioritize some updates over others; when more cognitive resources, time, or incentives are available, humans exhibit a greater propensity for cache updating [72–77]. For example, when given the opportunity to mentally simulate from their internal model during quiet rest, people show a greater degree of behavioral adaptation to task changes [78], consistent with offline cache updating. Furthermore, the degree of adaptation tracks neural measures of mental simulation [79].

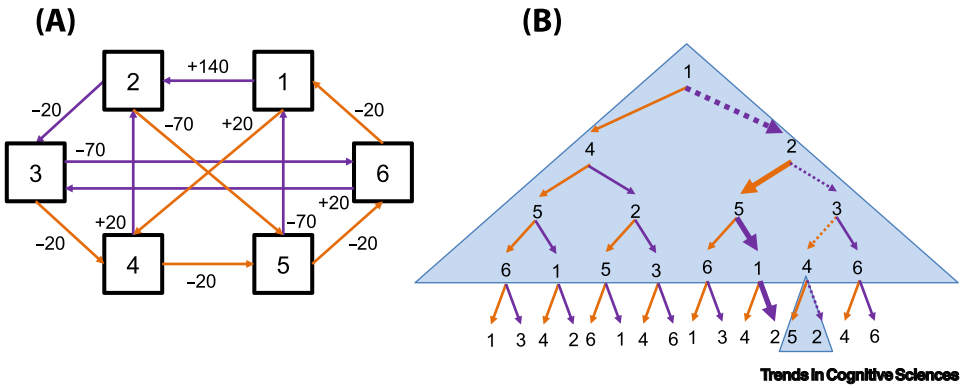

Figure 4. Planning Task Studied in Huys *et al.* [80]. (A) State transition diagram, arrows indicate different actions. The numbers next to the arrows indicate the reward collected on each transition. (B) Example of policy fragmentation. Starting in state 1, the optimal four-step action sequence is highlighted by the bold arrows. The blue triangles show a fragmentation of the decision tree consistent with subjects' choices (broken arrows), according to which the optimal solution to the three-step problem (1-2-3-4) is cached and then reused when computing the solution to the four-step problem.

Memoization can also provide insight into how old plans are reused to solve new problems. Huys and colleagues [80] analyzed human behavior in a planning task in which subjects selected a sequence of actions to maximize rewards, starting from a randomly chosen state on each trial (Figure 4A). Subjects did not in general choose the optimal path, instead relying on several heuristics. Most relevant for our purposes, they exhibited a bias towards paths that adhered to previously chosen subpaths (policy 'fragments'). For example, the optimal four-step path starting in state 1 is 1-2-5-1-2 (Figure 3B). However, subjects preferred the path 1-2-3-4-2. Borrowing ideas from earlier theoretical work [81,82], Huys and colleagues explained this bias as follows: subjects had already solved the optimal three-step path and they then reused this solution in solving the four-step path. This has the advantage of reducing the planning problem to a single-step choice, while still achieving a good (albeit suboptimal) solution. Consistent with the findings from mental arithmetic and imagery described earlier, the degree of fragment reuse increased across trials, ostensibly due to filling of the memo-table. Partition into fragments also carries similarities to the decomposition strategies used in these domains for partial reuse.

Policy fragmentation has kinship with a broader set of ideas about the role of memory in sequential decision making [83,84]. Of particular relevance is the work of Lengyel and Dayan [85], who proposed that entire sequences of actions are remembered as a unit and reused later when an agent occupies the same starting state. They argued that this form of 'episodic control' was particularly valuable when an agent has very little experience in a domain. In the low-data regime, agents cannot hope to build an accurate internal model of the world, nor can they hope to accurately estimate cached values by averaging samples, so episodic memories may be the agent's best bet (a claim supported by simulations and mathematical analysis). Recent work in artificial intelligence has combined this idea with the generalization permitted by neural networks to solve challenging sequential decision problems, demonstrating impressive gains in learning speed [86–88].

Concluding Remarks

Within psychology and neuroscience, memory has traditionally been studied as a goal in itself; memory tasks (e.g., list learning) explicitly study how information is stored and retrieved. In this review, we have argued for a conceptualization of memory as a ubiquitous participant in efficient computation. Similar ideas have historically been advocated for in cognitive science (Box 1), however, modern machine learning algorithms go beyond these approaches. These algorithms use powerful function approximation architectures to adaptively determine what information to store. They are also disciplined by domain knowledge (e.g., the Markov property and the Bellman equation) that provides principled constraints on the structure of memory.

The exercise of formalizing human cognition in computational terms has directly highlighted just how computationally expensive several of the tasks we naturally perform can be. This review highlights a crucial component of how intelligent behavior can arise despite limited computational power: by the use of memory as a computational resource. There remain several open questions in how this resource is deployed (see Outstanding Questions).

Outstanding Questions

What are the costs of computations, both the time required to perform new computations, as well as the space required to cache

How does the brain negotiate this space–time trade-off in choosing when to cache computations?

How are cached computations generalized to novel inputs?

How can we combine top-down and bottom-up memoization schemes to reap their complementary benefits?

What kinds of representations permit partial reuse by integrating information from cached values and new computation?

What are the memory systems (episodic, declarative, semantic, etc.) that underlie the caching of computations? How can we study their roles behaviorally and neurally?

Computational rationality or resource rationality posit another kind of trade-off, between accuracy and computational cost. What is the role of memory as a computational resource in this trade-off?

Acknowledgments

We are grateful to Tyler Brooke-Wilson, Timothy O'Donnell, Vikash Mansinghka, and Stuart Shieber for helpful comments. This work was supported by a gift from Microsoft Research, the Multi-University Research Initiative (ONR/DoD N00014-17-1-2961), and by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216.

References

1. Bentley, J.L. (1982) *Writing Efficient Programs*, Prentice-Hall
2. De Groot, A.D. (1946) *Het Denken van den Schaker [Thought and Choice in Chess]*, Noord Hollandsche
3. Newell, A. *et al.* (1958) Chess-playing programs and the problem of complexity. *IBM J. Res. Dev.* 2, 320–335
4. Logan, G.D. (1988) Toward an instance theory of automatization. *Psychol. Rev.* 95, 492

5. Anderson, J. (1987) Skill acquisition: compilation of weak-method problem solutions. *Psychol. Rev.* 94, 192–210
6. Laird, J.E. *et al.* (1986) Chunking in SOAR : the anatomy of a general learning mechanism. *Mach. Learn.* 1, 11–46
7. Michie, D. (1968) "Memo" functions and machine learning. *Nature* 218, 19–22
8. Pugh, W. and Teitelbaum, T. (1989) Incremental computation via function caching. In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of Programming Languages*, pp. 315–328
9. Acar, U.A. *et al.* (2003) Selective memoization. *ACM SIGPLAN Not.* 38, 14–25
10. Bird, R.S. (1980) Tabulation techniques for recursive programs. *ACM Comput. Surv. (CSUR)* 12, 403–417
11. Groen, G. and Parkman, J. (1972) A chronometric analysis of simple addition. *Psychol. Rev.* 79, 329–343
12. Ashcraft, M. and Battaglia, J. (1978) Cognitive arithmetic: evidence for retrieval and decision processes in mental addition. *J. Exp. Psychol. Hum. Learn. Mem.* 4, 527–538
13. Ashcraft, M.H. and Fierman, B.A. (1982) Mental addition in third, fourth, and sixth graders. *J. Exp. Child Psychol.* 33, 216–234
14. Carpenter, T.P. and Moser, J.M. (1984) The acquisition of addition and subtraction concepts in grades one through three. *J. Res. Math. Educ.* 15, 179–202
15. Logan, G. and Klapp, S. (1991) Automatizing alphabet arithmetic I. Is extended practice necessary to produce automaticity? *J. Exp. Psychol. Learn. Mem. Cogn.* 17, 179–195
16. Compton, B.J. and Logan, G.D. (1991) The transition from algorithm to retrieval in memory-based theories of automaticity. *Mem. Cogn.* 19, 151–158
17. Tarr, M.J. and Pinker, S. (1989) Mental rotation and orientation-dependence in shape recognition. *Cogn. Psychol.* 21, 233–282
18. Beishuizen, M. (1993) Mental strategies and materials or models for addition and subtraction up to 100 in Dutch second grades. *J. Res. Math. Educ.* 24, 294–323
19. Lucangeli, D. *et al.* (2003) Effective strategies for mental and written arithmetic calculation from the third to the fifth grade. *Educ. Psychol.* 23, 507–520
20. Lemaire, P. and Callies, S. (2009) Childrens strategies in complex arithmetic. *J. Exp. Child Psychol.* 103, 49–65
21. Shepard, R.N. and Metzler, J. (1971) Mental rotation of three-dimensional objects. *Science* 171, 701–703
22. Jolicoeur, P. (1985) The time to name disoriented natural objects. *Mem. Cogn.* 13, 289–303
23. Johansson, G. (1973) Visual perception of biological motion and a model for its analysis. *Percept. Psychophys.* 14, 201–211
24. Gershman, S.J. *et al.* (2016) Discovering hierarchical motion structure. *Vis. Res.* 126, 232–241
25. Xu, H. *et al.* (2017) Seeing "what" through "why": evidence from probing the causal structure of hierarchical motion. *J. Exp. Psychol. Gen.* 146, 896–909
26. Just, M.A. and Carpenter, P.A. (1985) Cognitive coordinate systems: accounts of mental rotation and individual differences in spatial ability. *Psychol. Rev.* 92, 137
27. Knill, D.C. and Richards, W. (1996) *Perception as Bayesian inference*, Cambridge University Press
28. Oaksford, M. and Chater, N. (2007) *Bayesian Rationality: The Probabilistic Approach to Human Reasoning*, Oxford University Press
29. Doya, K. *et al.* (2007) *Bayesian Brain: Probabilistic Approaches to Neural Coding*, MIT Press
30. Kording, K.P. *et al.* (2007) The dynamics of memory as a consequence of optimal adaptation to a changing body. *Nat. Neurosci.* 10, 779–786
31. Kruschke, J.K. (2008) Bayesian approaches to associative learning: from passive to active learning. *Learn. Behav.* 36, 210–226
32. Gershman, S.J. (2015) A unifying probabilistic view of associative learning. *PLoS Comput. Biol.* 11, e1004567
33. Dayan, P. and Kakade, S. (2001) Explaining away in weight space. In *Advances in Neural Information Processing Systems* (volume 13), pp. 451–457
34. Deneve, S. *et al.* (2007) Optimal sensorimotor integration in recurrent cortical networks: a neural implementation of Kalman filters. *J. Neurosci.* 27, 5744–5756
35. Gershman, S.J. (2017) Dopamine, inference, and uncertainty. *Neural Comput.* 29, 3311–3326
36. Doucet, A. *et al.* (2001) *Sequential Monte Carlo Methods in Practice*, Springer
37. Maddison, C.J. *et al.* (2017) Filtering variational objectives. In *Advances in Neural Information Processing Systems*, pp. 6573–6583
38. Saeedi, A. *et al.* (2017) Variational particle approximations. *J. Mach. Learn. Res.* 18, 2328–2356
39. Gu, S.S. *et al.* (2015) Neural adaptive sequential Monte Carlo. In *Advances in Neural Information Processing Systems*, pp. 2629–2637
40. Vul, E. *et al.* (2009) Explaining human multiple object tracking as resource-constrained approximate inference in a dynamic probabilistic model. In *Advances in Neural Information Processing Systems*, pp. 1955–1963
41. Levy, R.P. *et al.* (2009) Modeling the effects of memory on human online sentence processing with particle filters. In *Advances in Neural Information Processing Systems*, pp. 937–944
42. Kingma, D.P. and Welling, M. (2013) Auto-encoding variational Bayes. In *The 2nd International Conference on Learning Representations*
43. Hinton, G.E. *et al.* (1995) The "wake-sleep" algorithm for unsupervised neural networks. *Science* 268, 1158–1160
44. Rezende, D. and Mohamed, S. (2015) Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538
45. Paige, B. and Wood, F. (2016) Inference networks for sequential Monte Carlo in graphical models. In *International Conference on Machine Learning*, pp. 3040–3049
46. Ritchie, D. *et al.* (2016) Neurally-guided procedural models: amortized inference for procedural graphics programs using neural networks. In *Advances in Neural Information Processing Systems*, pp. 622–630
47. Marcus, G. (2018) Deep learning: A critical appraisal. *arXiv* Published online January 2, 2018. https://arxiv.org/abs/1801.00631
48. Lake, B.M. *et al.* (2017) Building machines that learn and think like people. *Behav. Brain Sci.* 40
49. Yildirim, I. *et al.* (2019) An integrative computational architecture for object-driven cortex. *Curr. Opin. Neurobiol.* 55, 73–81
50. Yildirim, I. *et al.* (2020) Efficient inverse graphics in biological face processing. *Sci. Adv.* 6, eaax5979
51. Dasgupta, I. *et al.* (2020) A theory of learning to infer. *Psychol. Rev.* 127, 412–441
52. Gershman, S.J. and Goodman, N. (2014) Amortized inference in probabilistic reasoning. In *Proceedings of the 36th Annual Conference of the Cognitive Science Society* (Bello, P. *et al.*, eds), pp. 517–522, Cognitive Science Society
53. Dasgupta, I. *et al.* (2018) Remembrance of inferences past: amortization in human hypothesis generation. *Cognition* 178, 67–81
54. Cohen, A.L. *et al.* (2017) Beliefs and Bayesian reasoning. *Psychon. Bull. Rev.* 24, 972–978
55. St. BT Evans, J. *et al.* (2002) Background beliefs in Bayesian inference. *Mem. Cogn.* 30, 179–190
56. Anderson, J. (1993) Problem solving and learning. *Am. Psychol.* 48, 35
57. Newell, A. *et al.* (1972) *Human problem solving*, Vol. 104. Prentice-Hall
58. Chase, W.G. and Simon, H.A. (1973) Perception in chess. *Cogn. Psychol.* 4, 55–81
59. van Opheusden, B. *et al.* (2017) A computational model for decision tree search. In *CogSci*
60. Kuperwajs, I. *et al.* (2019) Prospective planning and retrospective learning in a large-scale combinatorial game. In *2019 Conference on Cognitive Computational Neuroscience*, pp. 13–16
61. Bellman, R.E. (1957) *Dynamic Programming*, Dover
62. Sutton, R.S. (1988) Learning to predict by the methods of temporal differences. *Mach. Learn.* 3, 9–44
63. Mnih, V. *et al.* (2019) Playing *Atari* with deep reinforcement learning. *arXiv* Published online December 19, 2013. https://arxiv.org/abs/1312.5602
64. Mnih, V. *et al.* (2015) Human-level control through deep reinforcement learning. *Nature* 518, 529

65. Hamrick, J.B. *et al.* (2019) Combining q-learning and search with amortized value estimates. *arXiv* Published online December 5, 2019. https://arxiv.org/abs/1912.02807
66. Silver, D. *et al.* (2016) Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 484
67. Kool, W. *et al.* (2018 a) Competition and cooperation between multiple reinforcement learning systems. In *Goal-directed Decision Making*, pp. 153–178, Elsevier
68. Daw, N.D. *et al.* (2005) Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nat. Neurosci.* 8, 1704–1711
69. Daw, N.D. *et al.* (2011) Model-based influences on humans' choices and striatal prediction errors. *Neuron* 69, 1204–1215
70. Momennejad, I. *et al.* (2017) The successor representation in human reinforcement learning. *Nat. Hum. Behav.* 1, 680–692
71. Russek, E.M. *et al.* (2017) Predictive representations can link model-based reinforcement learning to model-free mechanisms. *PLoS Comput. Biol.* 13, e1005768
72. Mattar, M.G. and Daw, N.D. (2018) Prioritized memory access explains planning and hippocampal replay. *Nat. Neurosci.* 21, 1609–1617
73. Kool, W. *et al.* (2017) Cost-benefit arbitration between multiple reinforcement-learning systems. *Psychol. Sci.* 28, 1321–1333
74. Kool, W. *et al.* (2018) Planning complexity registers as a cost in metacontrol. *J. Cogn. Neurosci.* 30, 1391–1404
75. Keramati, M. *et al.* (2016) Adaptive integration of habits into depth-limited planning defines a habitual-goal–directed spectrum. *Proc. Natl. Acad. Sci.* 113, 12868–12873
76. Sezener, C.E. *et al.* (2019) Optimizing the depth and the direction of prospective planning using information values. *PLoS Comput. Biol.* 15, e1006827
77. Otto, R.A. *et al.* (2013) The curse of planning: dissecting multiple reinforcement-learning systems by taxing the central executive. *Psychol. Sci.* 24, 751–761
78. Gershman, S.J. *et al.* (2014) Retrospective revaluation in sequential decision making: a tale of two systems. *J. Exp. Psychol. Gen.* 143, 182–194
79. Momennejad, I. *et al.* (2018) Offline replay supports planning in human reinforcement learning. *eLife* 7, e32548
80. Huys, Q.J.M. *et al.* (2015) Interplay of approximate planning strategies. *Proc. Natl. Acad. Sci.* 112, 3098–3103
81. Wingate, D. *et al.* (2013) Compositional policy priors. In *MIT CSAIL Technical Report, 2013:007*
82. O'Donnell, T.J. (2015) *Productivity and Reuse in Language: A Theory of Linguistic Computation and Storage*, MIT Press
83. Gershman, S.J. and Daw, N.D. (2017) Reinforcement learning and episodic memory in humans and animals: an integrative framework. *Annu. Rev. Psychol.* 68, 101–128
84. Biderman, N. *et al.* (2020) What are memories for? The hippocampus bridges past experience with future decisions. *Trends Cogn. Sci.* 24, 542–556
85. Lengyel, M. and Dayan, P. (2007) Hippocampal contributions to control: the third way. In *Advances in Neural Information Processing Systems*, pp. 889–896
86. Blundell, C. *et al.* (2016) Model-free episodic control. *arXiv* Published online June 14, 2016. https://arxiv.org/abs/1606.04460
87. Pritzel, A. *et al.* (2017) Neural episodic control. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2827–2836, JMLR
88. Botvinick, M. *et al.* (2019) Reinforcement learning, fast and slow. *Trends Cogn. Sci.* 23, 408–422
89. Anderson, J. *et al.* (2004) An integrated theory of the mind. *Psychol. Rev.* 111, 1036–1060
90. Taatgen, N.A. and Lee, F.J. (2003) Production compilation: a simple mechanism to model complex skill acquisition. *Hum. Factors* 45, 61–76
91. Tambe, M. *et al.* (1990) The problem of expensive chunks and its solution by restricting expressiveness. *Mach. Learn.* 5, 299–348
92. Koller, D. and Friedman, N. (2009) *Probabilistic Graphical Models: Principles and Techniques*, MIT Press